SRM Storage and File Types

WLCG Data Management Coordination Group

Version 4
May 2, 2006

Overview

This document uses the definitions in the SRM v3 specification as a starting point for discussing the LHC experiment usage of SRM based storage system. It concerns itself with issues of file and cache copy lifetime as well as how the user may specific the "quality of storage" required in terms of number of copies of the file on disk and tape based storage.

Section 1 describes core type definitions as defined in SRM v3 while **Section 2** defines more concretely the difference between get and put operations within the specification with regards to file lifetimes.

Section 3 describes the subset of values for the core type definitions that will be used by LCG tools, and **Section 4** proposes two new type definitions for usage with the SRM specification, **Storage Class** and **Cache Attributes**, that the LHC experiments see as requirements in order to implement their data model.

1. Description of SRM Type Definitions

The following definitions are extracted verbatim from the SRM v3 specification (*http://sdm.lbl.gov/srm-wg/doc/v3/SRM.func.spec.v3.0.draft.pr1.html*). This is considered to be the reference definition of these terms and words when mentioned later in this document.

File Storage Types

In SRM v3, as of SRM v2.1, there are three file storage types which represent the lifetime of a file within the storage system. They have the following semantics:

- **Volatile**: This has an expiration time and the storage may delete all traces
- **Durable:** This has an expiration time, but the storage may not delete the file, and should raise an error condition instead.
- **Permanent**: This has no expiration time.

NOTE: A "file" in this sense equates directly with a SURL or a namespace entry within the storage system. It may have many different internal "copies" within the system (e.g. on tape, in disk pools), but deleting the file means deleting it from the storage namespace.

The File Storage Type does not imply anything with regard to access latency of the file, or the quality of service with regards to retention provided in storage system.

Retention Policies

Quality of Retention is a kind of Quality of Service. It acts as a notification to the storage system of the intention of the user regarding the “preciousness” of the file, and what would/could be done if the storage system lost the data.
There are three retention policies with the following semantics:

- **Replica**: Replica quality is appropriate for data that can be replaced since other copies can be accessed in a timely fashion.
- **Output**: Output quality is an intermediate level and is used for data which can be replaced by length or effort-full processes.
- **Custodial**: Custodial quality indicates that the data is in some sense unique, and all attempts should be made to ensure it is kept.

Access Latency Mode

This refers to how latency to access a file is improvable. Latency is changed by storage systems replicating a file internally on a storage area of different access latency.
Three access latency modes are defined, but in practice only the first two are used:

- **ONLINE**: This is the mode with the lowest latency, implying that there exists a copy of the file in the cache and available for the client
- **NEARLINE**: This represents file stored on a high latency medium, such as tape, which can have their latency automatically improved to ONLINE by a staging operation
- **OFFLINE**: A file in OFFLINE requires human intervention to achieve low latency.

Issuing a **PrepareToGet** call makes a file within the storage system ONLINE.

2. File Storage Type semantics on Get and Put operations

In the SRM specification, **PrepareToPut** and **PrepareToGet** are seen as symmetric operations, dealing with storing and retrieving SURLs from a storage system, and attributes in the operations are relevant to this SURL.

We believe that this leads to confusion, and there is an inherent asymmetry in the two operations:

- **PrepareToPut** is concerned with files within the storage system, represented by SURLs. A file has a file storage type associated with it.
- **PrepareToGet** is concerned with the cache copy associated with a particular SURL, in that we are asking for the storage system to provide us with a disk-resident cache copy of a file with some associated properties in terms of its accessibility.

In particular we believe that the requested lifetime and associated File Storage Type does not refer to the same thing; On a **PrepareToPut** operation, the storage type and lifetime apply to the SURL (or file within the storage system) and on a **PrepareToGet** operation lifetime applies to the requested cache copy of the file that is returned as a TURL.

If we consider that **PrepareToGet** is only concerned with the cache copy, it is not clear what the usage of a File Storage Type here is, since the lifetime is expressed numerically in the file lifetime. One suggestion (from Timur) is that the File Storage Type was used when using **PrepareToGet** to get a file from a remote SRM. This is now deprecated and not implemented, so it is possible that perhaps this argument could be removed from **PrepareToGet**.

3. LCG Usage of existing SRM parameters

The LHC experiments have expressed their wish to only use permanent files in storage systems – i.e. there should never be automatic deletion of files from the storage namespace. This means that all **PrepareToPut** operations will only use the **File Storage Type** parameter with value "Permanent". This is even true for "scratch" files, since the experiments will do their own bookkeeping, and clean up the files afterwards. This is done by the **srmRm** operation, which is a hard delete of the SURL that also removes it from the namespace.

The experiments also feel that there is no need to specify a File Storage Type when doing a **PrepareToGet**. All cache files are considered garbage collectable, and since it is possible for a lifetime is specified, this is enough.

Also, the experiments also feel that there is no supported use case for specifying File Storage Type on **PrepareToGet** in SRM v2.1 or v3, and perhaps this argument should be removed from the operation.

Retention Policies are seen to not be expressive enough, and will not be used by the LCG experiments. See below the proposed **Storage Class** which we believe is different expression of the same concept.

4. LCG proposed additions to the SRM Specification

Storage Classes

Currently the SRMv3 **PrepareToPut** does not allow any way for a given File Storage Class to specify Retention Quality. One possibility would be to require an extra argument of type **EnumRetentionQualityMode** (which already exists in the specification) which would specify the user requested retention quality.

Unfortunately, this does not seem to map well to LCG actual usage. We would suggest using a different concept instead, **Storage Class**, which maps directly into the minimum number of copies of a file that should be stored on Disk and Tape:

Storage Class	*Minimum Required Copies*	
	Tape	Disk
Tape0Disk1	0	1
Tape0DiskN	0	>1
Tape1Disk0	1	0
Tape1Disk1	1	1

Tape1DiskN	1	N
TapeNDisk0	>1	0
TapeNDisk1	>1	1
TapeNDiskN	>1	>1
…	…	…

Looking at the class **Tape1Disk0**, this means that we wish there to be always one copy on tape, and there is no requirement to have a copy permanently on disk – any copy that is created by the system can be garbage collected. **Tape1Disk1** is an extension of this, where we require the file to be "archived" on tape, usually for custodial reasons, but also for a copy to permanently reside on disk, for low latency access reasons.

We must accept that there will be short periods of time when perhaps the criteria is not met, especially when recovering from error, e.g. For a file in the **Tape1Disk1**, if the node holding the copy fails in an unrecoverable way, there would be a period while it is being staged back onto another node during which no disk copy would exist. This does not mean the Storage Class of the file changes during this period, merely a slightly reduced quality of service.

In order to change the Storage Class, this should be done by a separate (new) operation, **ChangeStorageClass**. This may have limited access on the storage system so that, for instance, only experiment production managers can change the Storage Class of a file.

Cache Attributes

Also, we would like to be able to specify extra properties for the TURL. Some of these are currently inferred by storage systems, e.g. the uid/DN/VO of the requesting user defines a VO-specific pool that they will be mapped into, the IP address of the requesting host can be used to put a file either on a pool for local access or for WAN access. Also some are currently in the SRM protocol, e.g. access Protocol and file lifetime.

We list below some TURL properties as a starting for discussion:

- Access Pattern (random/sequential)
- Access Speed (high/ low)
- Access Protocol (gridftp, rfio, dcap, …)

Also there are some properties then inherent to the node the file resides on that can affect the quality of service of access to the TURL.

- File System parameters (block size,…)
- TCP Transfer Protocol Properties (for WAN/LAN transfers)

We believe there should be a new parameter on the **PrepareToGet** and **PrepareToPut** operations that specific the attributes that are required for the TURL returned to get and put a file respectively.

[To be expanded with Olof's set of cache attributes…]